Amendments to the Specification:

Please insert the following paragraph after the paragraph ending on page 6, line 19:

E2 --Figure 5 is a flowchart showing the operation of the bundler.--

Please replace the paragraph beginning on page 9, line 3 with the following paragraph:

E3 --Figure 4 is a flow chart showing the steps required to implement the present invention, to emulate SSE instructions. First, a macroinstruction is decomposed into two or more microinstructions in a decompose function 310. The two microinstructions used to emulate the high half and the low half of the SSE operation are forced to issue in parallel and are dispatched simultaneously to the two FP units each capable of calculating two SP values per cycle, as shown in the emulation function 320. This is accomplished via a mechanism claimed in previously co-pending patent application, now issued U.S. Patent No. 6,618,801, entitled "Method and Apparatus for Implementing Two Architectures In a Chip Using Bundles That Contain Microinstructions and Template Information," inventors Knebel, et al., filed on the same date as this application, which is hereby incorporated by reference. Both micro-operations must be treated as a single operation. They move in lockstep with each other.--

Please insert the following new paragraphs after the paragraph ending on page 9, line 25:

--Bundler

E4 As noted above, the present invention operates by the use of a bundler. The bundler is part of the emulation engine. The emulation engine processes a sequence of operations (XUOPs). Between the emulation front end and the bundler is an XUOP queue, also referred to as an XUOP buffer. Within the emulation front end 0 is a microcode ROM (uROM). The uROM delivers information to the bundler. The function of the bundler is to take XUOPs and other information (including ZUOPs) delivered from the emulation front end within the emulation engine, converts this information into a valid 16-byte bundle as defined by the RISC ISA, and deliver to the execution engine two 16-byte bundles and associated pre-decode bits that can be decoded and executed in parallel without violating any architectural dependencies within the pair of bundles.

The emulation front end is required to deliver the following bits of information (referred to as "ZUOPs"), in addition to other information not described herein. These ZUOPs are to be used by the bundler as it creates the two 16-byte bundles.

EH

1. Syllable: 41-bit instruction that is understood by the execution engine.

2. Immediate: 32-bit immediate field that can be used as an operand.

3. Op-type: 3-bit field specifying which functional units can execute this type of Syllable.

4. Sub-Type: 3-bit field specifying further information specific to a particular Op-Type.

5. Bnd-Hint: 2-bit field indicating certain dependency restrictions between this Syllable, its predecessor and successor Syllables.

6. Reg-Valid: 4-bit field specifying whether each lf four separate fields within in the 41-bit Syllable contain valid register identifiers.

FIG. 5 shows the operation of the bundler in determining how many XUOPs to issue. The bundler issues either 0, 1, or 2 XUOPs per issue-group. The bundler attempts to issue two XUOPs at the same time, if possible. This determination is based on the number of XUOPs in the XUOP queue and on the application of certain rules, described below. The bundler must first determine how many entries are in the XUOP queue, in a determination function 200. If the XUOP queue has no entries, then the bundler outputs nothing, as shown by the no XUOP output function 210.

If the XUOP queue has one entry, then a determination function 220 determines whether the Bnd-Hint indicates that two XUOPs must be issued in parallel. If two XUOPs do not need to be issued in parallel, then the one XUOP in the XUOP queue is dispatched into two 16-byte bundles in the one XUOP output function 230. If the determination function 220 determines that two XUOPs must be issued in parallel, then the bundler outputs nothing in the no XUOP output function 210.

If the XUOP queue has two entries, then a determination function 240 determines whether the Bnd-Hint indicates that two XUOPs must be issued in parallel. If the determination function 240 determines that 2 XUOPs must be issued in parallel, then two XUOPs are dispatched into two 16-byte bundles in the two XUOP output function 250. If the determination function 240 determines that two XUOPs are not required to be issued in parallel, then the determination function 260 determines whether any of the following five rules apply:

1. A specific bit in a machine specific register is set to restrict dual issue.

2. Both XUOP's are destined for the same execution unit, unless they are both floating point operations or if they are both "general" ALU operations.

3. Both XUOP's have a Sub-Type that indicates they modify floating point (FP)-stack resources.

E4

4. Both XUOP's have a Sub-Type that indicates they could flush the pipeline based on a comparison result.

5. Comparing register fields that are indicated to be valid by the Reg-Valid bits shows that there is a register dependency hazard between two XUOP's.

If none of the five rules apply, then two XUOPs are dispatched into two 16-byte bundles in the two XUOP output function 250. If any of these five rules do apply, then one XUOP is dispatched into two 16-byte bundles, in the one XUOP output function 230.--